IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Alexander, III et al.**	§	
	§	Group Art Unit: **2192**
Serial No. **10/777,742**	§	
	§	Examiner: **Romano, John J.**
Filed: **February 12, 2004**	§	
	§	
For: **Method and Apparatus for Averaging Out Variations in Run-to-Run Path Data of a Computer Program**	§	

35525

PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on April 7, 2008.

A fee of $510.00 is required for filing an Appeal Brief. Please charge this fee to Yee & Associates Deposit Account No. 50-3157.

A one month extension of time is believed to be necessary. I authorize the Commissioner to charge the one month extension fee of $120.00 to Yee & Associates Deposit Account No. 50-3157. No additional extension of time is believed to be necessary. If, however, an additional extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to Yee & Associates Deposit Account No. 50-3157.

No additional extension of time is believed to be necessary. If, however, an additional extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

This appeal has no related proceedings or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

The claims in the application are: 1-23

B. STATUS OF ALL THE CLAIMS IN APPLICATION

Claims canceled: None

Claims withdrawn from consideration but not canceled: None

Claims pending: 1-23

Claims allowed: None

Claims rejected: 1-23

Claims objected to: None

C. CLAIMS ON APPEAL

The claims on appeal are: 1-23

STATUS OF AMENDMENTS

An Amendment after the Final Office Action of December 11, 2007, was not filed. Accordingly, the claims on appeal herein are as amended in the Response to Office Action filed September 17, 2007.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of claim 1 is directed to method, in a data processing system, for averaging out variations in trace data obtained from a plurality of executions of a computer program. Call tree data structures corresponding to the trace data for the plurality of executions of the computer program are obtained (**1620**, **Figure 16**; Specification, page 47, lines 23-25). The call tree data structures are added to generate an added call tree data structure (**1650**, **Figure 16**; Specification, page 47, line 25-page 48, line 8), and an average of values associated with each node in the added call tree data structure is calculated to generate an averaged call tree data structure (**1660**, **Figure 16**; Specification, page 48, lines 8-11). The averaged call tree data structure is outputted, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure (**1670 Figure 16**; Specification, page 48, lines 11-15).

B. CLAIM 9 - INDEPENDENT

The subject matter of claim 9 is directed to a computer program product that comprises a recordable-type computer readable medium having computer readable program code for averaging out variations in trace data obtained from a plurality of executions of a computer program. The computer program product has first instructions for obtaining call tree data structures corresponding to the trace data for the plurality of executions of the computer program (**1620**, **Figure 16**; Specification, page 47, lines 23-25). The computer program product also has second instructions for adding the call tree data structures to generate an added call tree data structure (**1650**, **Figure 16**; Specification, page 47, line 25-page 48, line 8), third instructions for calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure (**1660**, **Figure 16**; Specification, page 48, lines 8-11), and fourth instructions for outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure (**1670**, **Figure 16**; Specification, page 48, lines 11-15).

C. CLAIM 17 - INDEPENDENT

The subject matter of claim 17 is directed to an apparatus for averaging out variations in trace data obtained from a plurality of executions of a computer program. The apparatus has means (**302**, **304**, **Figure 3**; Specification, page 15, lines 12-13) for obtaining call tree data structures corresponding to the trace data for the plurality of executions of the computer program (**Figures 14A**, **14B**; Specification, page 44, lines 16-20). The apparatus also has means for adding the call tree data structures (**302**, **304**, **Figure 3**; Specification, page 15, lines 12-13) to generate an added call tree data structure (**Figure 14C**; Specification, page 47, lines 27-30), means for calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure (**302**, **304**, **Figure 3**; Specification, page 15, lines 12-13), and means for outputting the averaged call tree data structure (**306**, **308**, Figure 3; Specification, page 15, lines 12-15), wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure.

D. CLAIM 4 – DEPENDENT

The subject matter of claim 4, which depends from claim 1, specifies that adding the call tree data structures to generate an added call tree data structure includes copying a first call tree data structure, and walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure (**1930**, **Figure 19**; Specification, page 51, lines 23-27; also see page 43, line 20-page 44, line 6).

E. CLAIM 5 – DEPENDENT

The subject matter of claim 5, which depends from claim 4, specifies that walking the second call tree data structure over the first call tree data structure includes, for each node that exists in both the first call tree data structure and the second call tree data structure, generating a node in the added call tree data structure by adding a base value of the node in the second call tree data structure to a base value of a corresponding node in the first call tree data structure (**Figure 15**; Specification, page 45, line 19-page 46, line 16).

F. CLAIM 6 – DEPENDENT

The subject matter of claim 6, which depends from claim 4, specifies that walking the second call tree data structure over the first call tree data structure includes, for each node that exists in only one of the first call tree data structure and the second call tree data structure, creating a node in the added call tree data structure having a base value corresponding to the base value of the node that exists in either of the first call tree data structure or the second call tree data structure (**1930**, **Figure 19**; Specification, page 51, lines 23-27; also see page 43, line 20-page 44, line 6).

G. CLAIM 12 – DEPENDENT

The subject matter of claim 12, which depends from claim 9, specifies that the second instructions for adding the call tree data structures to generate an added call tree data structure includes instructions for copying a first call tree data structure, and instructions for walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure (**1930**, **Figure 19**; Specification, page 51, lines 23-27; also see page 43, line 20-page 44, line 6).

H. CLAIM 13 – DEPENDENT

The subject matter of claim 13, which depends from claim 12, specifies that the instructions for walking the second call tree data structure over the first call tree data structure includes, for each node that exists in both the first call tree data structure and the second call tree data structure, instructions for generating a node in the added call tree data structure by adding a base value of the node in the second call tree data structure to a base value of a corresponding node in the first call tree data structure (**Figure 15**; Specification, page 45, line 19-page 46, line 16).

I. CLAIM 14 – DEPENDENT

The subject matter of claim 14, which depends from claim 12, specifies that the instructions for walking the second call tree data structure over the first call tree data structure includes, for each node that exists in only one of the first call tree data structure and the second

call tree data structure, instructions for creating a node in the added call tree data structure having a base value corresponding to the base value of the node that exists in either of the first call tree data structure or the second call tree data structure (**1930**, **Figure 19**; Specification, page 51, lines 23-27; also see page 43, line 20-page 44, line 6).

J. CLAIM 20 – DEPENDENT

The subject matter of claim 20, which depends from 17, specifies that the means for adding the call tree data structures to generate an added call tree data structure includes means for copying a first call tree data structure, and means for walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure (**302**, **304**, **Figure 3**; Specification, page 15, lines 12-13).

K. CLAIM 21 – DEPENDENT

The subject matter of claim 21, which depends from claim 20, specifies that the means for walking the second call tree data structure over the first call tree data structure includes, for each node that exists in both the first call tree data structure and the second call tree data structure, means for generating a node in the added call tree data structure by adding a base value of the node in the second call tree data structure to a base value of a corresponding node in the first call tree data structure (**302**, **304**, **Figure 3**; Specification, page 15, lines 12-13)**; 1930**, **Figure 19**; specification, page 51, lines 23-27; also see page 43, line 20-page 44, line 6).

L. CLAIM 22 – DEPENDENT

The subject matter of claim 22, which depends from claim 20, specifies that the means for walking the second call tree data structure over the first call tree data structure includes, for each node that exists in only one of the first call tree data structure and the second call tree data structure, means for creating a node in the added call tree data structure having a base value corresponding to the base value of the node that exists in either of the first call tree data structure or the second call tree data structure (**302**, **304**, **Figure 3**; Specification, page 15, lines 12-13)**; 1930**, **Figure 19**; Specification, page 51, lines 23-27; also see page 43, line 20-page 44, line 6).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to review on appeal are as follows:

A. GROUND OF REJECTION 1

Claims 1, 4, 7, 9, 12, 15, 17, 20 and 23 stand finally rejected under 35 U.S.C. § 103 (a) as being unpatentable over Cohen et al., U.S. Patent No. 6,011,918, in view of Kazi et al., "JaViz: A client/server Java profiling tool."

B. GROUND OF REJECTION 2

Claims 2-3, 5-6, 8, 10-11, 13-14, 16, 18-19 and 21-22 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Cohen et al., U.S. Patent No. 6,011,918, in view of Kazi et al., "JaViz: A client/server Java profiling tool", and further in view of Alexander et al., "A unifying approach to perform analysis in the Java environment."

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1, 4, 7, 9, 12, 15, 17, 20 and 23)

Claims 1, 4, 7, 9, 12, 15, 17, 20 and 23 stand finally rejected under 35 U.S.C. § 103 (a) as being anticipated by Cohen et al., U.S. Patent No. 6,011,918 (hereinafter "Cohen"), in view of Kazi et al., "JaViz: A client/server Java profiling tool" (hereinafter "Kazi").

A.1. Claims 1, 4, 7, 9, 12, 15, 17, 20 and 23

In finally rejecting the claims, the Examiner states with respect to claim 1:

In regard to claim 1, **Cohen** discloses:

- *"A method, in a data processing system, for averaging out variations in trace data obtained from a plurality of executions of a computer program..."* (E.g., see Figure 5 & Column 12, lines 25-28), wherein each detailed trace file is analyzed to gathered in the merge step, comprising average execution time for each method.
- *". . . obtaining call tree data structures corresponding to the trace data for the plurality of executions of the computer program... "* (E.g., see Figure 8 & Column 10, lines 46-51), wherein call graphs with weighted nodes are disclosed.
- *". . . adding the call tree data structures to generate an added call tree data structure; calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure..."* (E.g., see Figure 5 & Column 12, lines 25-28), wherein profiles are collected for multiple runs and the results associated with each node of the call tree are averaged. It is also noted that the generated added call tree data structure, although not expressly disclosed or displayed, must be created in order to compute the average. The averaging of a metric for a respective node, is in a hierarchical structure, even if in intermediate textual format, and necessarily added and divided to obtain the average of the particular metric.

But **Cohen** does not expressly disclose "...the affect of variations in trace data of various executions of the computer program are minimized 'in the averaged call tree data structure". However, **Kazi** discloses:

- *". . . and outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the*

averaged call tree data structure." (E.g., see Table 1 & page 100, "Run-time statistics generation"), wherein to facilitate the performance analysis of the call graph, statistical information is averaged (execution times) and the standard deviation for each method, thereby minimizing the display for each execution.

Cohen and **Kazi** are analogous art because they are both concerned with the same field of endeavor, namely, a distributed application profiling tool. Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine **Kazi's** averaging minimization with **Cohen's** profiling method. The motivation to do so would have been to discover the amount of time spent in certain methods as disclosed by **Kazi** (See page 96, "Inefficient methods") to analyze the performance of the java application program.

Final Office Action dated December 11, 2007, pages 7-9.

Claim 1 on appeal herein is as follows:

1. A method, in a data processing system, for averaging out variations in trace data obtained from a plurality of executions of a computer program, comprising:

obtaining call tree data structures corresponding to the trace data for the plurality of executions of the computer program;

adding the call tree data structures to generate an added call tree data structure;

calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure; and

outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure.

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). For an invention to be *prima facie* obvious, the prior art must teach or suggest all claim limitations. *In re Royka,* 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974). In this case, the Examiner has not met this burden because all of the recited features of the claims are not found in the cited prior art references as believed by the Examiner. With respect to claim 1, for example, neither Cohen nor Kazi nor Cohen in view of Kazi discloses or suggests "adding the call tree data structures to generate an added call tree data structure",

"calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure", or "outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure".

Cohen is directed to a mechanism for generating client/server applications from an application written to execute on a single processing system. Cohen illustrates a call graph in Figure 4 wherein nodes of the call graph represent classes of an application, and subnodes correspond to programmed methods associated with each class. With reference to Figure 7, Cohen describes a process of assigning weights to the nodes (Fig. 6 of Cohen illustrates a call graph after weights have been assigned). The node weights are used to represent an approximation of client resource requirements such as, for example, the amount of RAM required to instantiate the object of a class (see, for example, column 10, lines 27-31 of Cohen).

The Examiner refers to Figure 5 and column 12, lines 25-28 of Cohen as disclosing "adding the call tree data structures to generate an added call tree data structure", and "calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure". Column 12, lines 18-36 is reproduced below for the convenience of the Board:

> Similarly, the computation of node and edge weights may be accomplished using dynamic information generated by profiling an exemplary execution of the program. In such an implementation profiling information rather than static analysis would be utilized to generated the weights. In a profiled implementation, the program is run such that it simulates a typical execution of the program, and profiling information is collected. Methods of collecting profiling information are well-known to those of skill in the art. Multiple runs can also be used, in which case, the results may be averaged. Based on the profiling information, node and edge weights may be assigned. In this case, node weights (single values or vectors) may be determined based on the actual resources consumed during the profiling run, optionally scaled by an uncertainty factor. Edge weights are assigned based on the actual number of calls made from one method to another and the volume of data passed during those calls. The remainder of the partitioning process may then be carried out as described above.

The above recitation states simply that "Multiple runs can also be used, in which case, the results may be averaged". Cohen does not disclose, either in the above recitation or elsewhere in the patent, how the averaging is achieved, and certainly does not describe or in any way suggest a

method for averaging variations in trace data that includes "adding the call tree data structures to generate an added call tree data structure" and "calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure" as is recited in claim 1.

Furthermore, Cohen does not disclose or suggest that the averaging referred to in the above recitation is with respect to call tree data structures. Claim 1 recites that call tree data structures corresponding to trace data for a plurality of executions of the computer program are obtained, and that the adding and calculating steps are performed with respect to the call tree data structures. Cohen appears to disclose only that profiling information is averaged, and, as stated in the above-reproduced portion of Cohen, "Based on the profiling information, node and edge weights may be assigned". There is nothing in Cohen to suggest that call tree data structures are added to generate an added call tree data structure, or that an average of values associated with each node in an added call tree data structure are calculated to generate an averaged call tree data structure. Only the present application contains such a disclosure.

In responding to Appellants' arguments that Cohen does not disclose or suggest "adding the call tree data structures to generate an added call tree data structure", the Examiner asserts that "Cohen's generated call tree data structure averages the values of the respective nodes, wherein the averaging of a metric/value for a respective node, necessarily comprises taking the sum of a group of values and dividing by the number of values" (see page 3 of Final Office Action dated December 11, 2007). Appellants respectfully disagree. A disclosure of averaging profiling information and assigning node and edge weights based on the [averaged] profiling information is not a teaching of "adding the call tree data structures to generate an added call tree data structure" as recited in claim 1. Again Cohen does not describe how averaging takes place, nor does the reference specify what aspects of the profiling information are averaged.

In responding to Appellants' arguments that Cohen does not disclose or suggest "calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure", the Examiner states that "[i]t is also noted that the generated call tree data structure, although not expressly discloses or displayed, must be created in order to compute the average, even if only in intermediate textual format" (see sentence bridging pages 3 and 4 of Final Office Action dated December 11, 2007). Appellants again respectfully disagree. As indicated above, Cohen does not disclose adding call tree data

structures to generate an added call tree data structure, and thus, cannot disclose or suggest "calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure."

For at least all the above reasons, Cohen does not disclose or suggest the steps of "adding the call tree data structures to generate an added call tree data structure", or "calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure" as recited in claim 1.

Kazi does not supply the above described deficiencies in Cohen, nor does the Examiner assert otherwise. The Examiner cites Kazi as disclosing "outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure." Appellants respectfully disagree. As noted by the Examiner, Kazi states, on page 100 thereof "To facilitate the performance analysis of the call graph, statistical information about each of the methods in the program is gathered in the merge step." The merge step is described beginning on page 99 of Kazi and involves merging trace files to produce "one detailed trace for each jvm that contains all of the relevant information, including the client/server activity."

At best, accordingly, Kazi may disclose merging traces to provide a detailed trace to facilitate performance analysis. Such a disclosure, however, is not a teaching or suggestion of outputting or otherwise providing an averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure". To the contrary, Kazi provides a merged trace to provide more detailed information, not to minimize the effect of variations in various executions of a computer program.

In responding to Appellants' arguments that Kazi does not disclose or suggest "outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure", the Examiner states:

> In regard to the argument that *Kazi* does not teach minimizing the effect of variations in various executions of a computer program (See response, page 13, 2nd

paragraph,) Kazi teaches merging the trace data. In regard to the trace data, Kazi discloses:

> 'To facilitate the performance analysis of the call graph, statistical information about each of the methods in the program is gathered in the merge step. Each detailed .prf trace file is analyzed to gather the total number of calls made to each method, the maximum, minimum, and average execution times, and the standard deviation of the execution time for each method." (Kazi, page 100, "run-time statistics generation – emphasis added.)

Final Office Action dated December 11, 2007, pages 4-5.

The above disclosure in Kazi of gathering a total number of calls, or gathering maximum, minimum and average execution times is not the same as outputting an "averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure. Kazi does not minimize variations in trace data of various executions and does not generate an averaged call tree data structure.

Therefore, Kazi does not disclose or suggest "outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure", and does not supply the deficiencies in Cohen.

For at least all the above reasons, neither Cohen nor Kazi nor Cohen in view of Kazi discloses or suggests "adding the call tree data structures to generate an added call tree data structure", "calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure", or "outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure" as recited in claim 1, and the Examiner has not established a *prima facie* case of obviousness in rejecting claim 1. Claim 1, accordingly, patentably distinguishes over the references in its present form.

Independent claims 9 and 17 recite similar subject matter as claim 1 and also patentably distinguish over Cohen in view of Kazi for similar reasons as discussed above with respect to claim 1.

Claims 4, 7, 12, 15, 20 and 23 depend from and further restrict one of the independent

claims, and also patentably distinguish over the cited references, at least by virtue of their dependency.

A.2. Claims 4, 12 and 20

Claim 4 depends from and further restricts claim 1 and is as follows:

> 4. The method of claim 1, wherein adding the call tree data structures to generate an added call tree data structure includes:
> copying a first call tree data structure; and
> walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure.

In finally rejecting claim 4, the Examiner states:

> In regard to claim **4**, the rejections of base claim 1 are incorporated. Furthermore, **Kazi** discloses:
> - *". . . walking a second call tree data structure over the first call tree data structure to generate the added call tree data structures."* (E.g., see Table 1 & page 111, "visualizer"), wherein traversing (walking) the nodes to gather the pertinent information is disclosed.

Final Office Action dated December 11, 2007, page 9.

Appellants respectfully submit that neither Cohen nor Kazi nor Cohen in view of Kazi discloses or suggests wherein adding the call tree data structures to generate an added call tree data structure includes "copying a first call tree data structure", and "walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure."

Initially, Appellants respectfully submit that the references do not disclose or suggest "copying a first call tree data structure." The Examiner does not address this step in rejecting the claim. Perhaps, the Examiner is alleging (silently) that this limitation is disclosed in Cohen. Appellants, however, are unable to identify any disclosure or suggestion in Cohen of copying a call tree structure for any reason. Similarly, Kazi contains no such disclosure or suggestion of copying a call tree data structure. Therefore, the Examiner has not established a *prima facie* case of obviousness in rejecting claim 4, for at least this reason.

In addition, the references also do not disclose or suggest "walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure." Kazi, at

best, discloses traversing nodes to gather pertinent information. This is not the same as "walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure" as recited in claim 4. As described, for example, on page 51, lines 23-27 of the present specification, "[t]he call tree data structures are then walked over each other in order to generate a minimized call tree data structure that includes only those nodes that are common to each call tree data structure (step **1930**)." Kazi does not disclose or suggest "walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure", and claim 4 patentably distinguishes over the references for this reason as well.

For at least all the above reasons, the Examiner has not established a *prima facie* case of obviousness in rejecting claim 4 as being obvious over Cohen in view of Kazi, and claim 4 patentably distinguishes over the cited references in its own right as well as by virtue of its dependency from claim 1.

Claims 12 and 20 recite similar subject matter as claim 4 and also patentably distinguish over the cited art in their own right as well as by virtue of their dependency.

For at least all the above reasons, claims 1, 4, 7, 9, 12, 15, 17, 20 and 23 patentably distinguish over Cohen in view of Kazi, and it is respectfully requested that the Board reverse the Examiner's Final Rejection of those claims.

B. GROUND OF REJECTION 2 (Claims 2-3, 5-6, 8, 10-11, 13-14, 16, 18-19 and 21-22)

Claims 2-3, 5-6, 8, 10-11, 13-14, 16, 18-19 and 21-22 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Cohen in view of Kazi, and further in view of Alexander et al., "A unifying approach to perform analysis in the Java environment" (hereinafter "Alexander").

B.1. Claims 2-3, 5-6, 8, 10-11, 13-14, 16, 18-19 and 21-22

Claims 2-3, 5-6, 8, 10-11, 13-14, 16, 18-19 and 21-22 depend from and further restrict one of independent claims 1, 9 and 17. Alexander does not supply the deficiencies in Cohen and Kazi as discussed in detail above with respect to the independent claims. Therefore, claims 2-3, 5-6, 8, 10-11, 13-14, 16, 18-19 and 21-22 patentably distinguish over Cohen in view of Kazi and further in view of Alexander, at least by virtue of their dependency.

B.2. Claims 5, 6, 13, 14, 21 and 22

Claims 5 and 6 depend from and further restrict claim 4 and are as follows:

5. The method of claim 4, wherein walking the second call tree data structure over the first call tree data structure includes:

for each node that exists in both the first call tree data structure and the second call tree data structure, generating a node in the added call tree data structure by adding a base value of the node in the second call tree data structure to a base value of a corresponding node in the first call tree data structure.

6. The method of claim 4, wherein walking the second call tree data structure over the first call tree data structure includes:

for each node that exists in only one of the first call tree data structure and the second call tree data structure, creating a node in the added call tree data structure having a base value corresponding to the base value of the node that exists in either of the first call tree data structure or the second call tree data structure.

Appellants respectfully submit that neither Cohen nor Kazi nor Alexander nor their combination discloses or suggests "generating a node in the added call tree data structure by adding a base value of the node in the second call tree data structure to a base value of a corresponding node in the first call tree data structure" as recited in claim 5, or "creating a node in the added call tree data structure having a base value corresponding to the base value of the node that exists in either of the first call tree data structure or the second call tree data structure" as recited in claim 6.

In finally rejecting claim 5, the Examiner states:

> In regard to claim **5**, the rejections of base claim 4 are incorporated. **Cohen** teaches the adding of values associated with each node as disclosed above with relation to claim **1**. However, **Cohen** and **Kazi** do not expressly disclose "...adding a base value of the node in the second call tree data structure to a base value of a corresponding node in the first call tree data structure." However, **Alexander** discloses:
>
> - *"...a base value ...."* (E.g., see page 124, first paragraph), wherein the base, calls and cum values are disclosed.
>
> Therefore, it would have been obvious to add the base value of the second node tree data structure to a base value of a corresponding

node in the first tree data structure to generate the average for each node metric as disclosed above.

Final Office Action dated December 11, 2007, page 11.

Alexander related to performance analysis in a Java environment. The complete paragraph bridging pages 123 and 124 of Alexander is reproduced below for the convenience of the Board:

> The xarc report is organized in paragraphs, or stanzas. Stanzas are separated by dashed (= =) lines. Each stanza includes a record for "self", a set of "parent" records, and a set of "child" records. Every unique consumer is represented by a stanza in which the self record identifies that consumer by name in the "function" column. Three key types of metrics are reported for each consumer: "calls," "base," and "cum" (short for cumulative). Calls indicates the number of times that a consumption event (e.g., a function call) has been recorded on behalf of this consumer. Base and cum both pertain to the quantity of resource consumed. Base indicates the amount consumed directly by this consumer in the context defined by its set of parents. Cum indicates the amount consumed both by the consumer directly and all of its children. (The examples illustrate a variant of the arcflow reports that normalize the resource consumption to percentages rather than show the absolute counts.) Also, there is special treatment of recursion, which will not be discussed here.

Nowhere, either in the above reproduced portion or elsewhere in Alexander, is there any disclosure or suggestion of generating a node in an added call tree data structure "by adding a base value of the node in the second call tree data structure to a base value of a corresponding node in the first call tree data structure. The base, call and cum metrics described in Alexander are not the same as the base values of nodes referred to in claim 5, and the reference certainly does not disclose or suggest adding base values of nodes in first and second call tree data structures in connection with walking the second call tree data structure over the first call tree data structure.

Claim 5, accordingly, and corresponding claims 13 and 21 patentably distinguish over Cohen in view of Kazi and Alexander in their own right as well as by virtue of their dependency.

In rejecting claim 6, the Examiner states:

> In regard to claim **6**, the rejections of base claim **4** are incorporated. But **Cohen** and **Kazi** do not expressly disclose "...for each node that exists in only one of the first call tree data structure and the second call tree data structure, creating a node in the added call tree data structure having a base value corresponding to the base value of the node that exists in either of the first call tree data structure or the second call tree data structure.". However, it would have been an inherent result of the averaging computation if a value only existed for one node.

Final Office Action dated December 11, 2007, page 11.

Appellants respectfully disagree that "creating a node in the added call tree data structure having a base value corresponding to the base value of the node that exists in either of the first call tree data structure or the second call tree data structure" is an "inherent result" of walking the second call tree data structure over the first call tree data structure, nor has the Examiner provided any explanation as to why such a procedure would be inherent. As discussed above, the references do not disclose or suggest walking a second call tree data structure over the first call tree data structure to generate an added call tree data structure as recited in claim 4, and certainly do not disclose or suggest the procedure for doing so recited in claim 6. Claim 6, accordingly, and corresponding claims 14 and 22 also patentably distinguish over the cited references in their own right as well as by virtue of their dependency.

For at least all the above reasons, claims 2-3, 5-6, 8, 10-11, 13-14, 16, 18-19 and 21-22 patentably distinguish over Cohen in view of Kazi and Alexander, and it is respectfully requested that the Board reverse the Examiner's Final Rejection of those claims.

/Gerald H. Glanzman/

Gerald H. Glanzman
Reg. No. 25,035
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal is as follows:

1. A method, in a data processing system, for averaging out variations in trace data obtained from a plurality of executions of a computer program, comprising:

obtaining call tree data structures corresponding to the trace data for the plurality of executions of the computer program;

adding the call tree data structures to generate an added call tree data structure;

calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure; and

outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure.

2. The method of claim 1, further comprising:

inputting the trace data to an arcflow tool, wherein the arcflow tool generates the call tree data structures based on the trace data.

3. The method of claim 1, wherein the call tree data structures are xtree data structures.

4. The method of claim 1, wherein adding the call tree data structures to generate an added call tree data structure includes:

copying a first call tree data structure; and

walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure.

5. The method of claim 4, wherein walking the second call tree data structure over the first call tree data structure includes:

for each node that exists in both the first call tree data structure and the second call tree data structure, generating a node in the added call tree data structure by adding a base value of the node in the second call tree data structure to a base value of a corresponding node in the first call tree data structure.

6. The method of claim 4, wherein walking the second call tree data structure over the first call tree data structure includes:

for each node that exists in only one of the first call tree data structure and the second call tree data structure, creating a node in the added call tree data structure having a base value corresponding to the base value of the node that exists in either of the first call tree data structure or the second call tree data structure.

7. The method of claim 1, wherein calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure includes:

dividing values associated with each node in the added call tree data structure by a total number of call tree data structures that were added to generate the added call tree data structure.

8. The method of claim 1, wherein the values associated with each node include a base value, a number of calls, a cumulative value, and an absolute cumulative value.

9. A computer program product, comprising:

a recordable-type computer readable medium having computer readable program code for averaging out variations in trace data obtained from a plurality of executions of a computer program, the computer program product comprising:

first instructions for obtaining call tree data structures corresponding to the trace data for the plurality of executions of the computer program;

second instructions for adding the call tree data structures to generate an added call tree data structure;

third instructions for calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure; and

fourth instructions for outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure.

10. The computer program product of claim 9, further comprising:

fifth instructions for inputting the trace data to an arcflow tool, wherein the arcflow tool generates the call tree data structures based on the trace data.

11. The computer program product of claim 9, wherein the call tree data structures are xtree data structures.

12. The computer program product of claim 9, wherein the second instructions for adding the call tree data structures to generate an added call tree data structure include:

instructions for copying a first call tree data structure; and

instructions for walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure.

13. The computer program product of claim 12, wherein the instructions for walking the second call tree data structure over the first call tree data structure include:

for each node that exists in both the first call tree data structure and the second call tree data structure, instructions for generating a node in the added call tree data structure by adding a base value of the node in the second call tree data structure to a base value of a corresponding node in the first call tree data structure.

14. The computer program product of claim 12, wherein the instructions for walking the second call tree data structure over the first call tree data structure includes:

for each node that exists in only one of the first call tree data structure and the second call

tree data structure, instructions for creating a node in the added call tree data structure having a base value corresponding to the base value of the node that exists in either of the first call tree data structure or the second call tree data structure.

15. The computer program product of claim 9, wherein the third instructions for calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure include:

instructions for dividing values associated with each node in the added call tree data structure by a total number of call tree data structures that were added to generate the added call tree data structure.

16. The computer program product of claim 9, wherein the values associated with each node include a base value, a number of calls, a cumulative value, and an absolute cumulative value.

17. An apparatus for averaging out variations in trace data obtained from a plurality of executions of a computer program, comprising:

means for obtaining call tree data structures corresponding to the trace data for the plurality of executions of the computer program;

means for adding the call tree data structures to generate an added call tree data structure;

means for calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure; and

means for outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure.

18. The apparatus of claim 17, further comprising:

means for inputting the trace data to an arcflow tool, wherein the arcflow tool generates the call tree data structures based on the trace data.

19. The apparatus of claim 17, wherein the call tree data structures are xtree data structures.

20. The apparatus of claim 17, wherein the means for adding the call tree data structures to generate an added call tree data structure includes:

means for copying a first call tree data structure; and

means for walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure.

21. The apparatus of claim 20, wherein the means for walking the second call tree data structure over the first call tree data structure includes:

for each node that exists in both the first call tree data structure and the second call tree data structure, means for generating a node in the added call tree data structure by adding a base value of the node in the second call tree data structure to a base value of a corresponding node in the first call tree data structure.

22. The apparatus of claim 20, wherein the means for walking the second call tree data structure over the first call tree data structure includes:

for each node that exists in only one of the first call tree data structure and the second call tree data structure, means for creating a node in the added call tree data structure having a base value corresponding to the base value of the node that exists in either of the first call tree data structure or the second call tree data structure.

23. The apparatus of claim 17, wherein the means for calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure includes:

means for dividing values associated with each node in the added call tree data structure by a total number of call tree data structures that were added to generate the added call tree data structure.

EVIDENCE APPENDIX

This appeal brief presents no additional evidence.

RELATED PROCEEDINGS APPENDIX

This appeal has no related proceedings.